

Figure 1 - Verilog-AMS Circuit Example - Prior Art

```
1 // divider from Verilog-ams standarization committee public circuits
2 `timescale 10ns/1ns
3 `include "disciplines.h"
4 `include "connect.h"
5
6 module top;
7     reg clk;
8     wire sys_clk;
9
10    // digital constructs
11    initial clk = 0;
12    always #5 clk = ~clk;          // 1uS Clock Generator
13    assign sys_clk = clk;
14
15    // instantiations
16    zdetect my_dev(sys_clk, divout);
17    lpf #(tau(1.59e-8)) tenMlpf(divout, tenMout);
18 endmodule
19
20 module zdetect(in,out);
21     input in;
22     output out;
23     electrical in,out;
24     integer n, state;
25     parameter div = 5;
26
27     // analog blocks code analog circuits as equations
28     analog begin
29         @(cross(V(in) - 2.5, +1)) n = n + 1;
30         if (n >= div )begin
31             if (state == 0) state = 1;
32             else state = 0;
33             n = 0;
34         end
35         V(out) <+ state * 5;
36     end endmodule
37
38 module lpf(in, out);
39     inout in, out;
40     electrical in, out;
41     parameter real tau = 1e-3;
42
43     analog
44     begin
45         V(out) <+ laplace_nd(V(in), {1.0}, {1.0, tau});
46     end
47 endmodule
```

Fig. 2

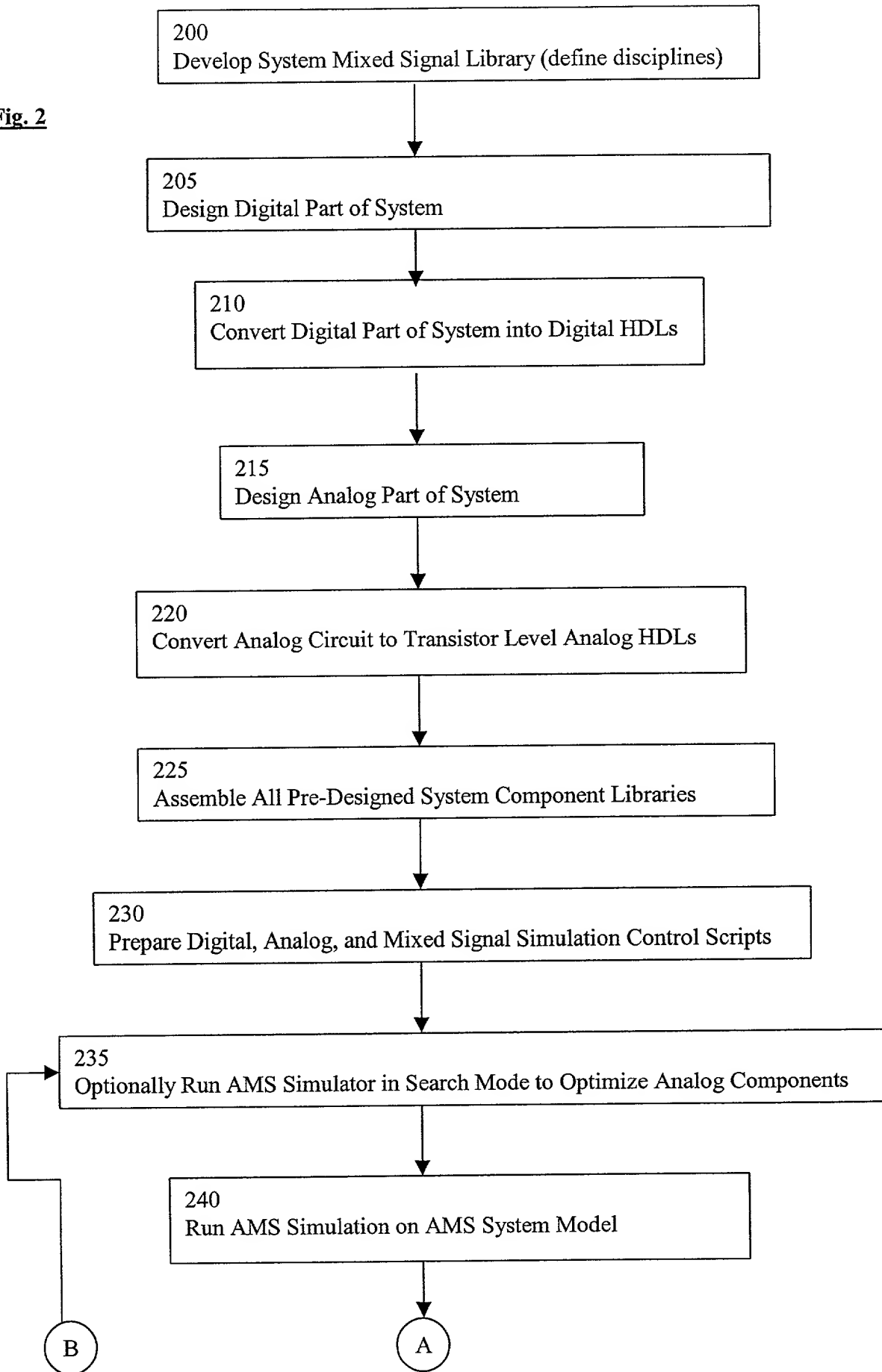
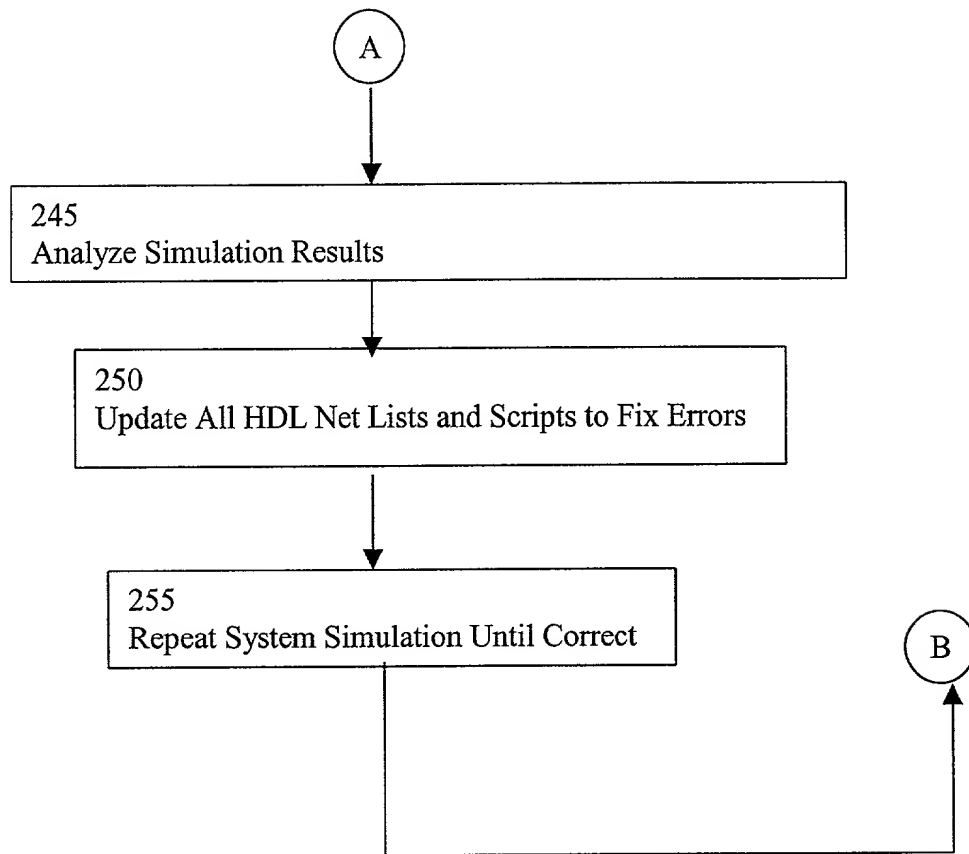


Fig. 2 (cont'd)



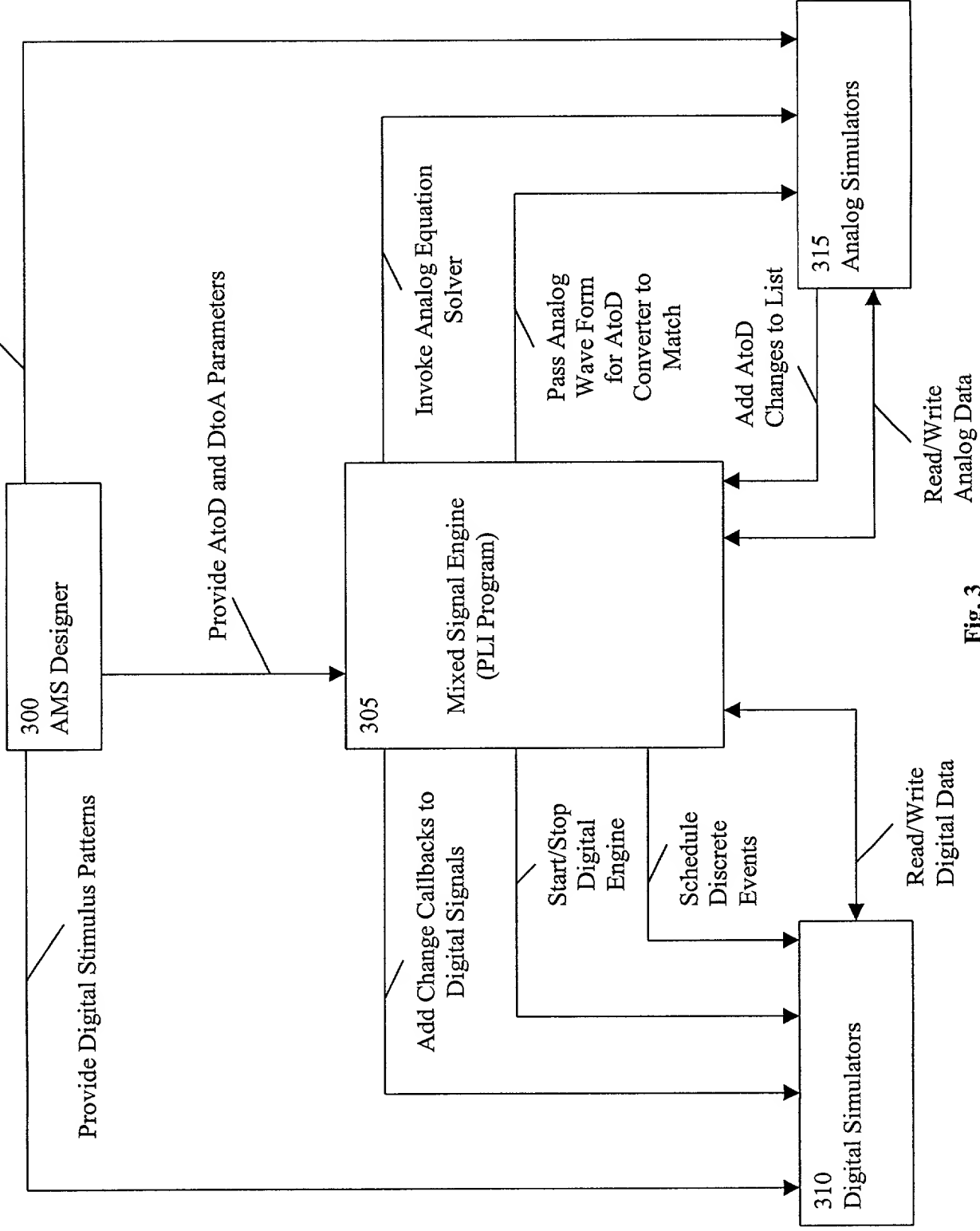


Fig. 3

Fig. 4

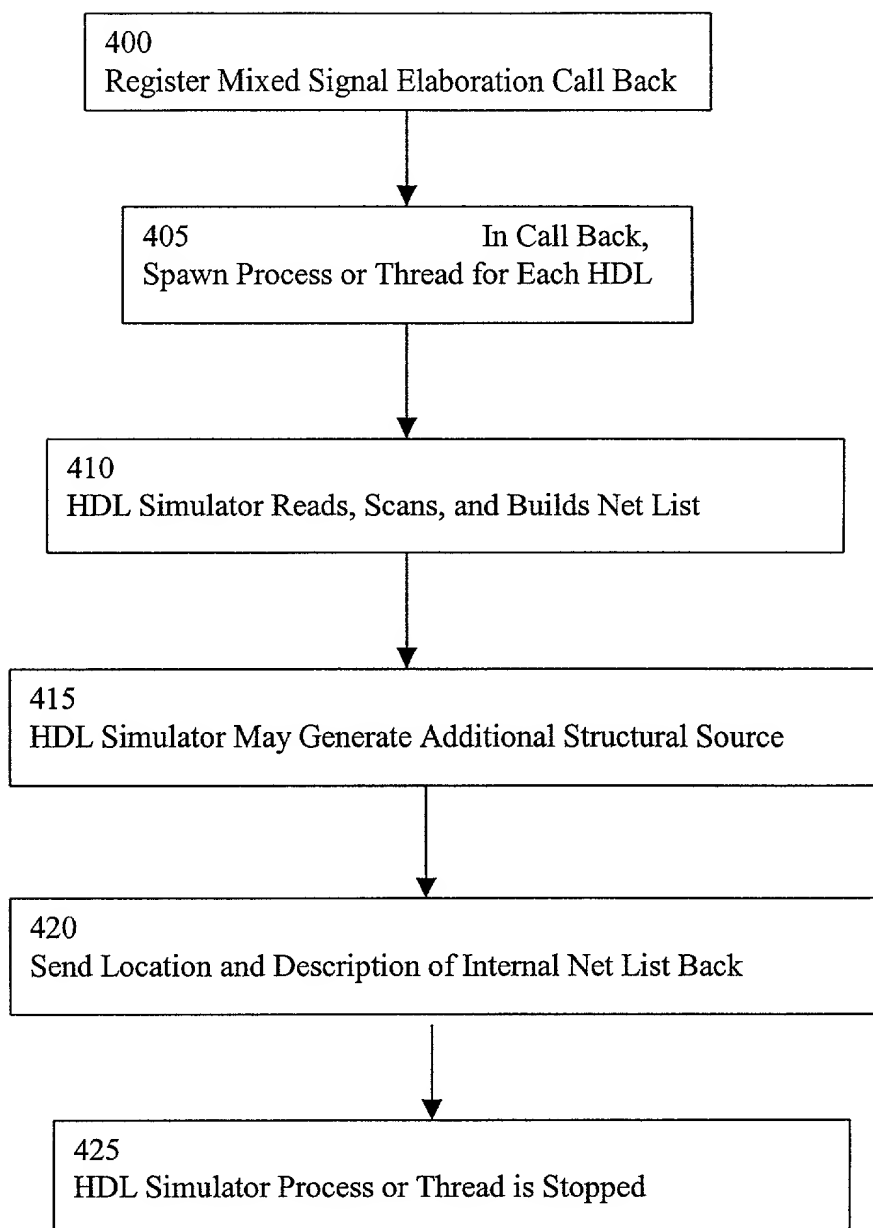


Fig. 5

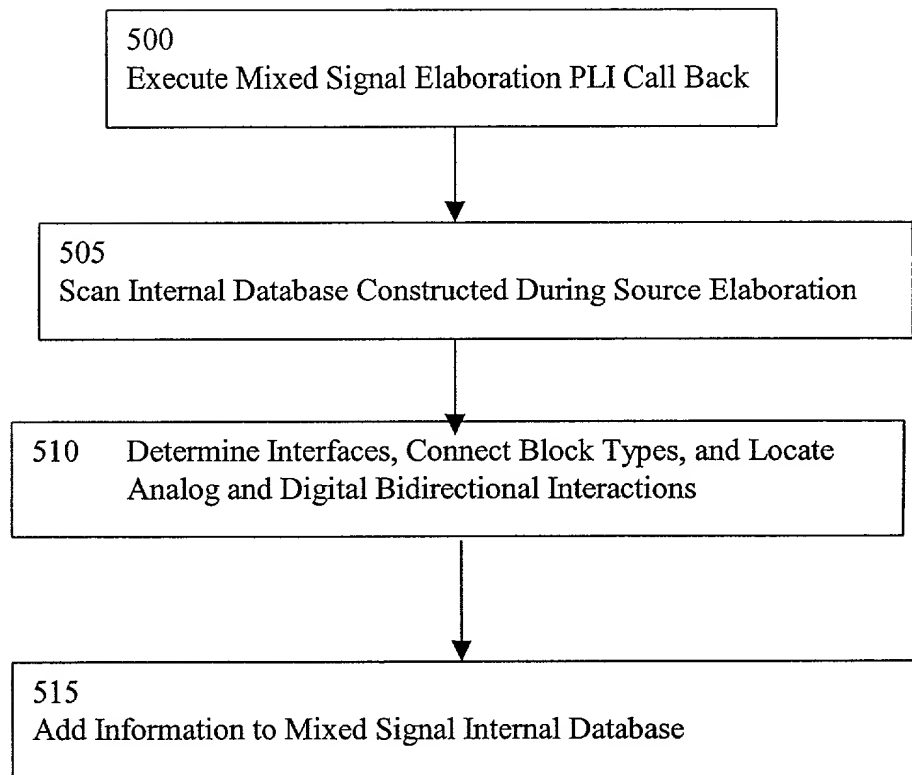


Fig. 6

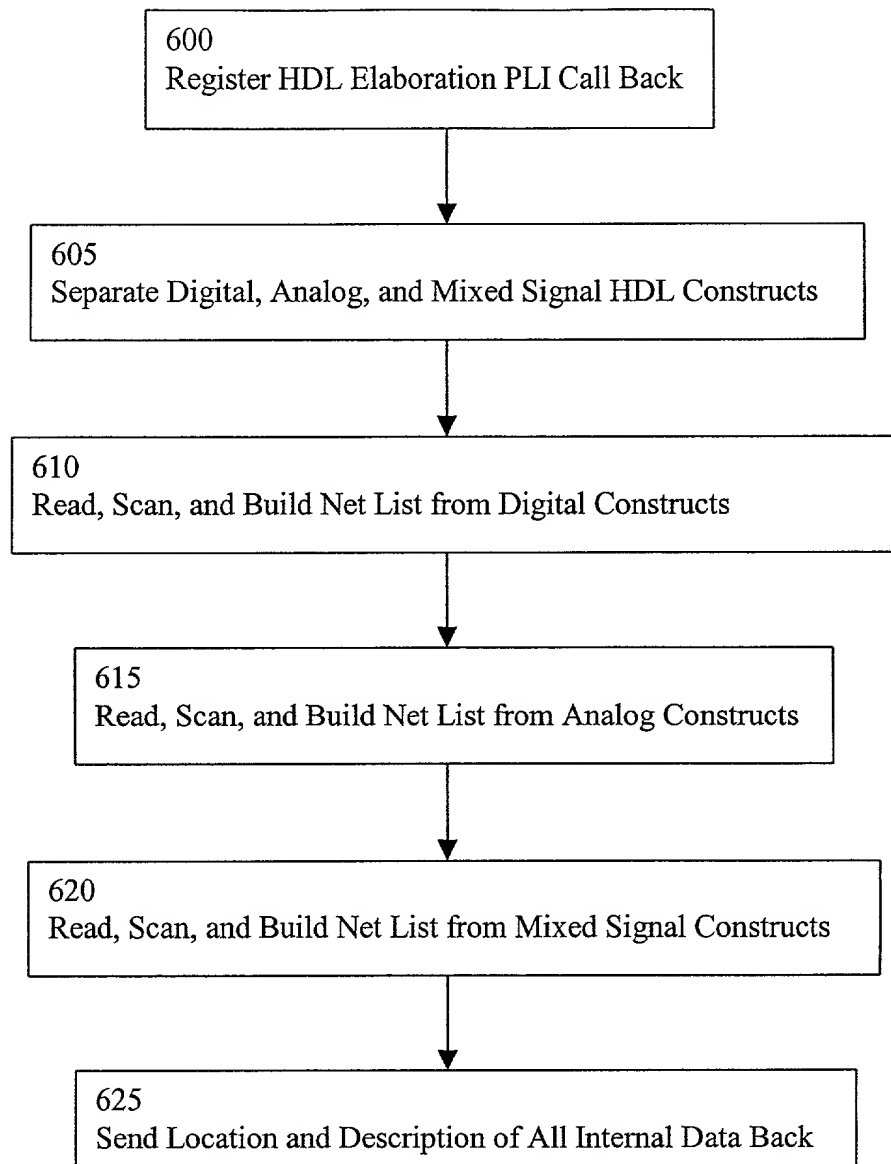


Fig. 7

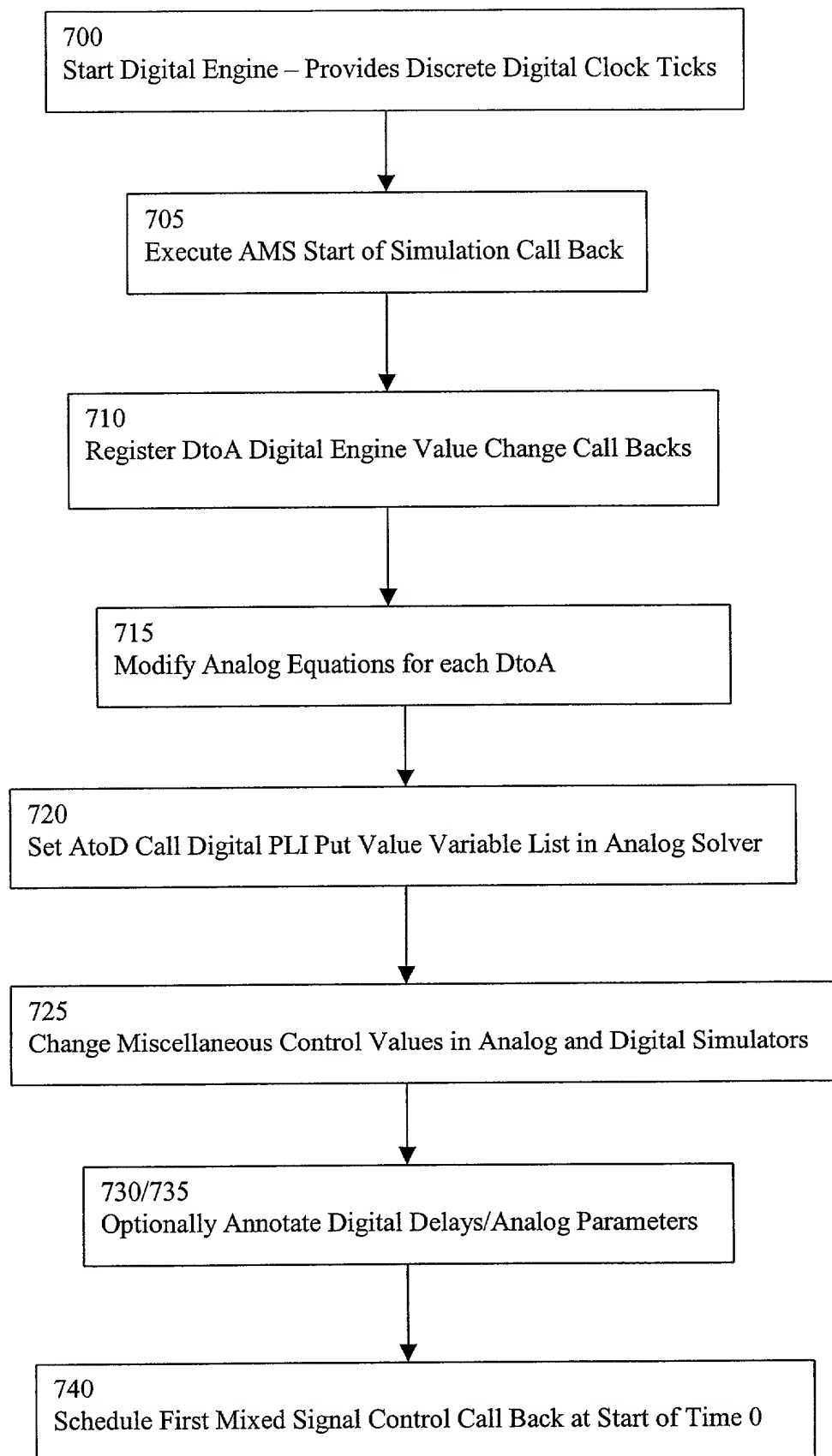


Fig. 8

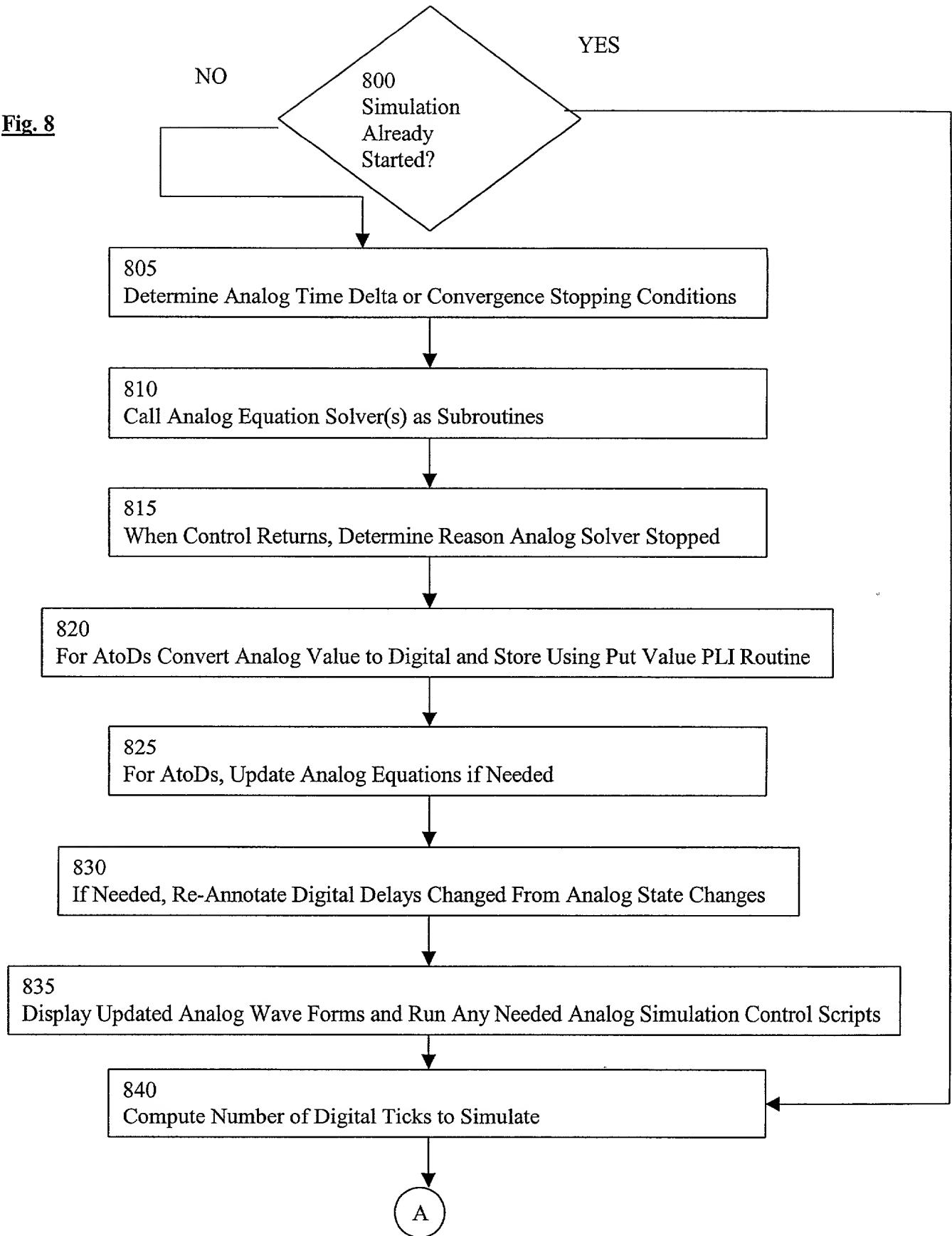
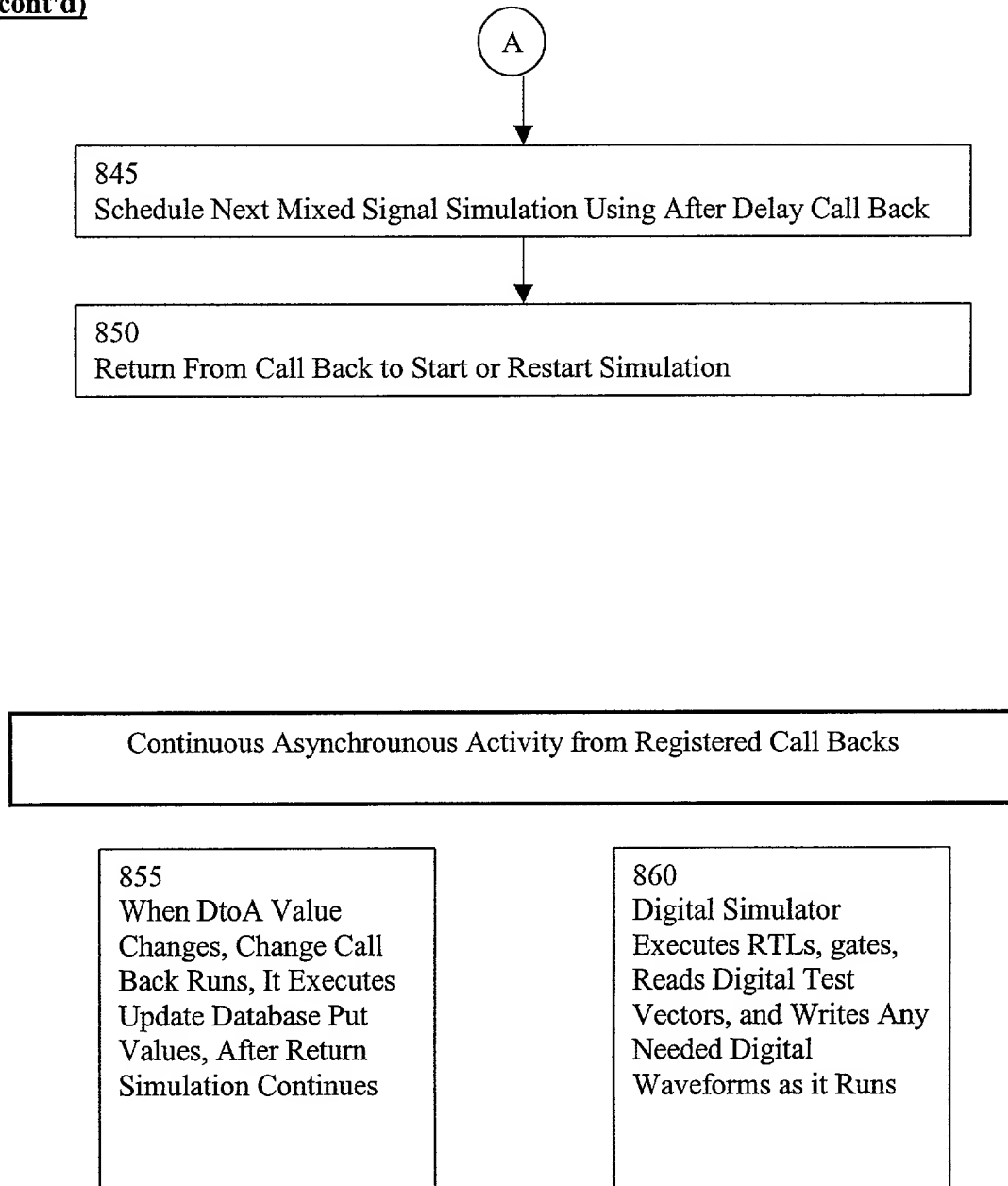


Fig. 8 (cont'd)



9.2.4 The synchronization loop

The digital and analog kernels shall be synchronized so neither computes results which the other is ineligible to accept. The synchronization algorithm can exploit characteristics of the analog and digital kernels described in the next section. A sample run is shown in Figure 9-4.

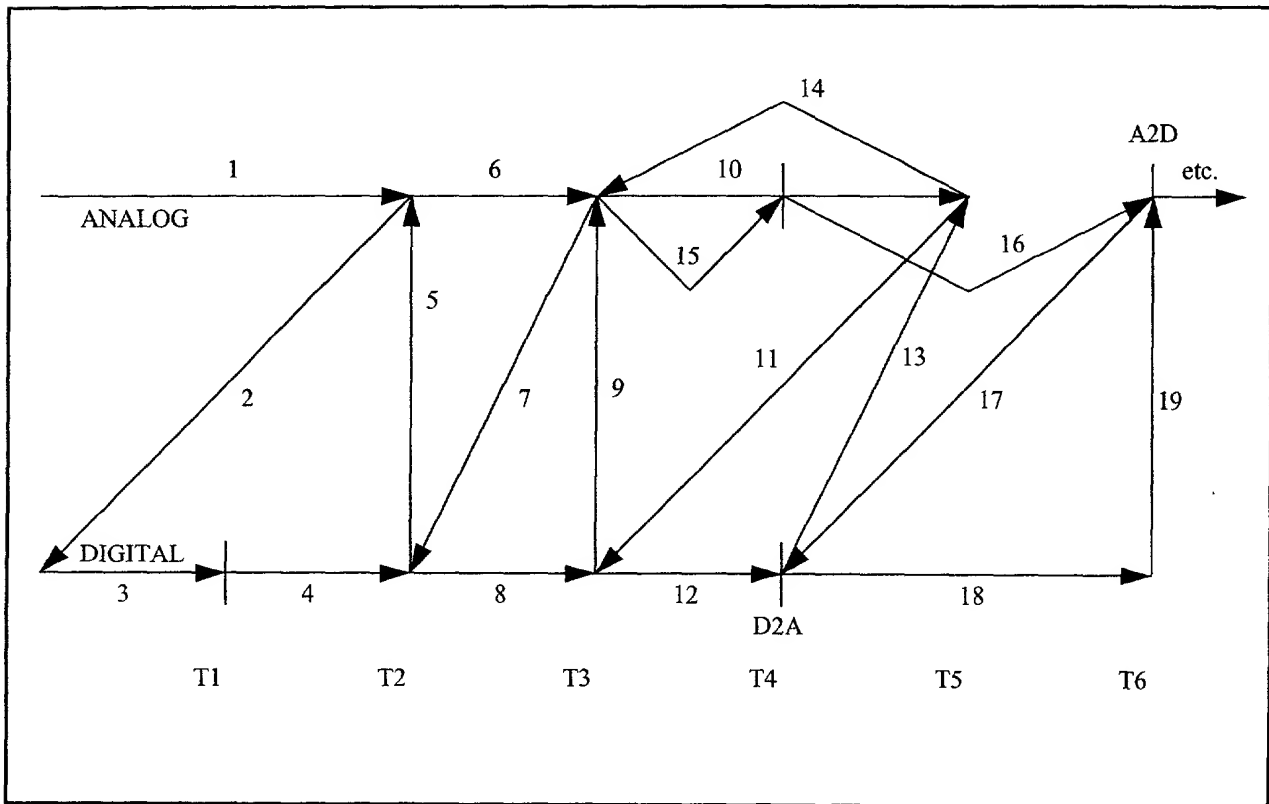


Figure 9-4 Sample run

1. The Analog engine begins transient analysis and sends state information to the Digital engine (1,2).
2. The Digital engine begins to run using its own time steps (3); however, if there is no D2A event, the Analog engine is not notified and the digital engine continues to simulate until it can not advance its time without surpassing the time of the analog solution (4). Control of the simulation is then returned to the analog engine (5). This process is repeated (7,8,9,10, and 11).
3. If the Digital engine produces a D2A event (12), control of the simulation is returned to the Analog engine (13). The analog engine returns to the point at which the digital engine last surrendered control (14). The Analog engine recalculates the analog solution up to the time when the D2A event occurred (15). The Analog engine then takes the next time step (16).

FIG. 9 (1 of 3)

4. If the Analog engine produces an A2D event, it returns control to the Digital engine (17), which simulates up to the time of the A2D event and then surrenders control (18 and 19).
5. This process continues until transient analysis is complete.

9.2.5 Assumptions about the analog and digital algorithms

1. Advance of time in a digital algorithm
 - The digital simulation has some minimum time granularity and all digital events occur at a time which is some integer multiple of that granularity.
 - The digital simulator can always accept events for a given simulation time provided it has not yet executed events for a later time. Once it executes events for a given time, it can not accept events for an earlier time.
 - The digital simulator can always report the time of the most recently executed event and the time of the next pending event.
2. Advance of time in an analog algorithm
 - The analog simulator advances time by calculating a sequence of solutions. Each solution has an associated time which, unlike the digital time, is not constrained to a particular minimum granularity.
 - The analog simulator can not tell for certain the time when the next solution converges. Thus, it can tell the time of the most recently calculated solution, but not the time of the next solution.
 - In general, the analog solution is a function of one or more previous solutions. Having calculated the solution for a given time, the analog simulator can either accept or reject that solution; it can not calculate a solution for a future time until it has accepted the solution for the current time.
3. Analog to digital events
 - Analog to digital events are generated by conversion elements (which are analog/digital behavioral models) when evaluated by the analog simulator.
 - Analog events (e.g., `cross`, `initial_step`, and `final_step`) cause an analog solution of the time where they occur.
 - Thus, any analog to digital event is generated as the result of a particular transient solution. This means events can stay associated with the solution which produced them until they are passed to the digital simulator, then they can be rejected along with the solution if it is rejected.

FIG. 9 (2 OF 3)

4. Digital to analog events shall cause an analog solution of the time where they occur.

Figure 9 (3 of 3)

Fig. 9 (3 of 3)

Fig. 10

